Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5175754, 14 pages
https://doi.org/10.1155/2022/5175754

Research Article

An Efficient User's Attribute Revocation Scheme Suitable for Data Outsourcing in Cloud Storage

Fang Zheng,[1] **Xinguang Peng**,[2] **and Zhidan Li**[3]

[1]*Shanxi University of Finance and Economics, Taiyuan, China*
[2]*Taiyuan University of Technology, Taiyuan, China*
[3]*Beijing University of Posts and Telecommunications, Beijing, China*

Correspondence should be addressed to Fang Zheng; alice20110324@126.com

Received 20 July 2021; Revised 8 December 2021; Accepted 22 December 2021; Published 15 February 2022

Academic Editor: Omprakash Kaiwartya

With the advances of cloud data centers and cloud service, many application scenarios are developed such as enterprise business, the Vehicular Ad Hoc Networks (VANET), Sensor network. Confidentiality and efficiency are two key problems. CP-ABE is one of fine-grained access control cryptographic technologies and it is widely applied in outsourced data in cloud storage to protect the user's privacy. In addition, besides confidentiality, computational cost is an important factor which makes the application of CP-ABE possible in the situations. In this paper, we propose a novel and fast scheme based on CP-ABE algorithm from the respect of the user's attribute revocation to make it faster. In this scheme, we construct an identical tree T2 that has the same structure as the actual access control tree T1. When the user issues to request the encrypted text preserved on CSP, CSP first tries to decrypt CT2 which is encrypted according to the access policy based on the identical tree T2 and also preserved on it, with the subset of the secret keys corresponding to the indexes of the attributes provided by the user. If CSP can successfully decrypt it, it proves that he/she has the authorization to access the cryptographic text CT1 and CSP will send CT1 to the user. Otherwise, CSP recognizes that the user has no access to CT1 and rejects to send CT1 to the user. Namely, we can decide if we authorize the right to access CT1 by judging if the secret key provided by the user can decrypt CT2. Because CT1 and CT2 have the same access control structure, they have the same secret key for decryption; that is, when some attributes are revoked, if the secret key cannot decrypt CT2, it cannot decrypt CT1. While CSP can directly decrypt CT2, but not CT1, CSP can judge if the user has access to CT1 after the attribute revocation by decryption of CT2. Moreover, we propose the construction method of CT1 and CT2. Finally, we prove that the scheme is secure and viable.

1. Introduction

With the development of cloud services, many application scenarios based on cloud data centers appear, such as enterprise business, the Vehicular Ad Hoc Networks (VANET), and Sensor network. Meanwhile these services require confidentiality and efficiency.

As one of fine-grained access control technologies, CP-ABE (Ciphertext-Policy Attribute-Based Encryption) is widely applied in the outsourced data in cloud storage to protect the user's privacy and prevent CSP (Cloud Service Provider) from peeping the user's contents. But it meets a problem about revocation which includes three kinds [1–5]. One is system attributes revocation which denotes that one or more attributes on access control tree is revoked including the attributes being removed or the logic relationship among attributes being destroyed; that is, it means the access control tree is changed. The second is user revocation, which denotes that the user loses the access to the cryptographic text. The third is user's attributes revocation, which means that the user's partial attributes are revoked; that is, AA (Attribute Authorization) revokes the user's partial attributes which probably makes him/her lose the right to visit the ciphertext. For example, assume that A is one of teachers in university B and A has the attributes of Prof (Professor), *Dr (Doctor)*, and *CD (Computer Department)*. In this system, B takes the role of AA and the three attributes of A are authorized by B and B can revoke one of the attributes,

such as *Dr*. Then the attributes owned by *A* have been changed from *{Prof, Dr, CD}* to *{Prof, CD}*. Suppose that the access control tree is as in Figure 1 and ciphertext *C* is encrypted according to the access control policy. Then when the attribute *CD* is revoked, the user will lose the right to visit ciphertext *C*, while if the revoked attribute is *Dr*, then the user still has the access to *C*. So, we can conclude that the user's attributes revocation can influence him/her to visit the ciphertext.

There are many literature studies on the three cases. When all the attributes of the user are revoked, we will think that the user been revoked. In this paper, we will discuss the last two cases, that is, the user revocation and user's attributes revocation.

To the situations which require the high time efficiency, for example, the Vehicular Ad Hoc Networks (VANET) and Sensor network, a fast CP-ABE algorithm is very important. The existing schemes of user's attribute revocation are mainly realized by reencrypting the ciphertext [6–8] or updating the private keys of the nonrevoked users and reencrypting the ciphertext. Our scheme neither updates the secret keys nor reencrypts the ciphertext. We accomplish the attribute revocation and user revocation by constructing a tree $T2$ identical with the real access control tree $T1$ for every user. The two trees have the same complete nodes and structure. The leaf nodes denote the different attributes. To a fixed system, its system tree with all the attributes provided by the system is fixed. So, we can define the secret keys set for the system tree as $SK = \{ski | i = 1, 2, \ldots, n\}$ and the attributes set as $S = \{si | i = 1, 2, \ldots, n\}$ and the leaf nodes set as $LF = \{xi | i = 1, 2, \ldots, n\}$; then there is one-to-one correspondence between the elements in *SK* and *S* and there is one-to-one correspondence between the elements in *S* and *LF*. When the system tree is generated, *SK*, *S*, and *LF* will be generated and fixed. To the different users, they have different leaf nodes which are authorized by AA. Figure 2(a) is a system tree which has four leaf nodes and the whole attributes set is $S = \{A1, A2, A3, A4, A5\}$, while Figures 2(b) and 2(c) show two trees, T1 and T2, which both belong to the same user and the attributes set is $S1 = S2 = \{A1, A3, A4\}$. From the figure, we can know that the tree corresponding to any user is a subtree whose leaf nodes are a subset of the system tree and the attributes set of any user is a subset of the whole attributes set of the system. So, the secret keys based on the subtree are the subset of those based on the system tree if they are all generated by the same curve parameters. To one user, suppose that his/her secret keys are set as $SK1$ and $SK2$ which are, respectively, generated according to trees $T1$ and $T2$ which have the same structure. If $SK1$ can decrypt the ciphertext encrypted based on the access control tree $T1$, then the cryptographic text encrypted based on tree $T2$ can also be decrypted by $SK2$ which are from the same leaf nodes as $SK1$. According to this principle, we can know whether the nonrevoked attributes set of the user has the right to decrypt the ciphertext $CT1$ by judging if *CT2* encrypted based on the identical tree $T2$ can be decrypted with the secret keys on the corresponding leaf nodes. Although $SK1$ and $SK2$ are generated from the same leaves of trees that have the same structure, $SK1$ and $SK2$ are not the same if the parameters that generate their elliptic curves are different. So, $SK2$ is generated by and preserved on *CSP*; in the same way, ciphertext $CT2$ is encrypted by and preserved on *CSP*. *CSP* can decide if the user has access to $CT1$ which is encrypted by the user itself and uploaded on *CSP* by judging if $CT2$ can be decrypted by $SK2$. $SK2$ varies with the variation of the leaf nodes on the access control tree, which is noticed by uploading the new leaf node indexes of tree $T1$ to CSP.

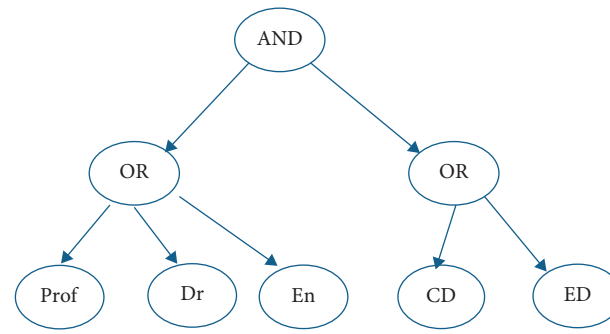

Figure 1: The example of CP-ABE.

The contributions of this paper are as follows:

(1) Our scheme is novel. We construct a tree $T2$ that is identical with the access control tree $T1$. By judging whether the identical tree $T2$ can be decrypted by $SK2$, CSPs decide whether to send ciphertext $CT1$ to the users.

(2) Our scheme is fast. The secret keys- (*SK*-) based system trees are shared by all the users and only generated once. To the different user, the secret key $SK2$ based on $T2$ is a subset of *SK*. It is not necessary to regenerate $SK2$ but select the elements from SK to reconstruct *SK2*, which saves the elapsed time.

(3) Our scheme is secure. Trees $T1$ and $T2$ have different curve parameters. So, the secret keys of $T2$ cannot decrypt ciphertext encrypted based on $T1$.Therefore, CSP cannot decrypt the data owner's cryptographic text.

(4) Our scheme does not need to update the secret keys and ciphertext and suits the scenarios that require high time efficiency, such as the Vehicular Ad Hoc Networks (VANET) and IoT network.

The remainder of the paper is organized as follows: Section 2 presents related work; Section 3 discusses preliminaries of cryptography; Section 4 gives the definition of our scheme; Section 5 discusses the detailed realization of our scheme; Section 6 presents the secure analysis; Section 7 presents the time efficiency analysis; Section 8 gives optimization scheme; Section 9 discusses the experiment; Section 10 gives the conclusion.

2. Related Work

2.1. Green Energy and Time Efficiency in Cloud Data Centers. Green energy and time efficiency in cloud data centers are two hot spots, which can keep the applications setup on it

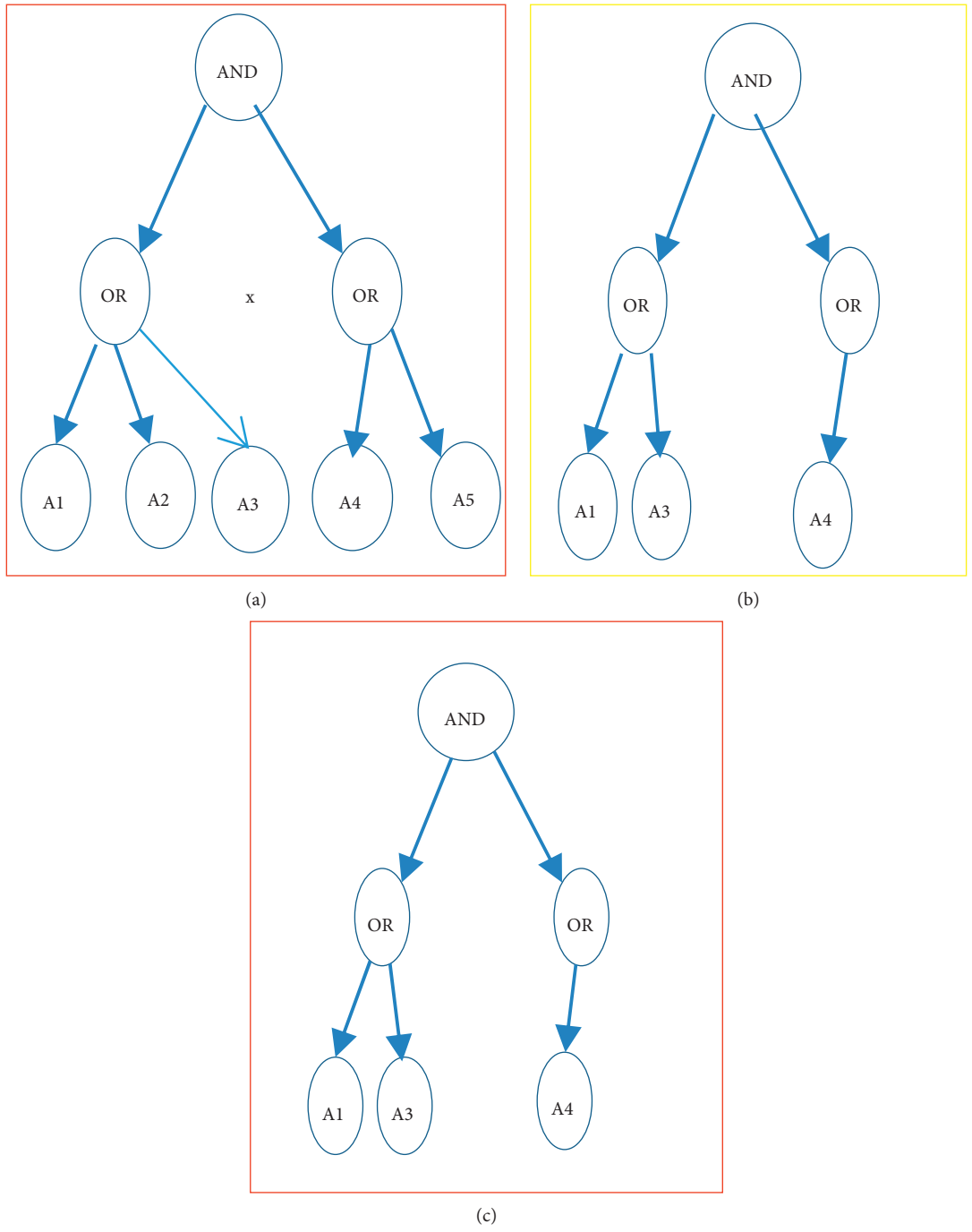

Figure 2: (a) The system tree to a fixed system. (b) The access control policy tree T1 corresponding to a fixed user. (c) The identical tree T2 corresponding to a fixed user.

running efficiently. The existing literature focus on the energy saving, resource distribution, overload reduction, algorithm improvement, and so forth.

Mohammed Joda Usman et al. in 2019 [9] proposed a scheme to solve the problem of distributing virtual resources in cloud data centers with an Energy-oriented Flower Pollination Algorithm (E-FPA), which minimizes resource wastage and increases energy efficiency. Jitendra Kumar Verma et al. [10] proposed three heuristic models combining the relation between power consumption and resource utilization and virtual machine technique to reduce energy consumption with minimal variation in service-level agreements negotiated. Sourya Joyee et al. in 2015 [11] proposed an efficient encryption based on CP-ABE by dividing the encryption process into two phases, offline phase and online phase, as well as a fast decryption scheme by introducing a third proxy server to partially decrypt the ciphertext and then transmit the rest to the mobile device to

compensate the shortcoming of the slow speed of the mobile device. The works in [9, 10] kept the cloud servers running efficiently from the aspect of energy leverage. The work in [11] improved the algorithm to reduce the overload of the cloud server. Compared with the previously mentioned schemes, our methods boost the efficiency of cloud server from the improvement of user's attribute revocation.

2.2. CP-ABE Algorithm. CP-ABE is a classical public key encryption algorithm based on ciphertext attributions widely used on many scenarios, especially in cloud services. In this section, we will introduce the study trajectory.

Yu et al. in 2012 [12] proposed an attribute revocation scheme, the idea of which is that when the attributes are revoked, AA reencrypts the ciphertexts and updates the private keys for all the nonrevoked users. To the situation where a lot of attributes or users are revoked, the scheme cannot satisfy the need. Hur and Noh in 2011 [13] and Xie et al. in 2013 [14] proposed the user and attributes revocation scheme based on the attribute key encryption keys (KEKs) generated by a global binary tree in which the leaf nodes denote the users. Naruse et al. in 2014 [15] proposed an attribute revocation scheme to CP-ABE schemes using a cloud server by proxy reencryption to revoke attributes. This scheme does not require generation of the new secret key when granting attributes to a user. In this scheme, the ciphertext needs to be updated. Cheng et al. in 2013 [16] proposed an optimization revocation scheme based on CP-ABE by selecting a random partial block instead of the whole file to perform revocation to decrease the cost of revocation. But the overhead of data publishing and retrieval is increased. Genlang et al. in 2019 [17] proposed a revocation scheme that constructs an AA tree in which every attribute is a child node of the root and has many leaf nodes named accumulators representing a user. One user has one or more attributes. When the user requests to visit the ciphertext, CSPs will generate the reencryption key to encrypt the ciphertext for the user with the nonrevoked attributes. This scheme needs to reencrypt the ciphertext. Shan-shan Tu et al. in 2012 [18] proposed a fully fine-grained direct attribute revocation scheme based on composite order bilinear groups and LSSS including all revoked user lists corresponding to every attribute. CSPs need to reencrypt the ciphertext. Benjamin Wesolowski et al. in 2016 [19] proposed a ciphertext-policy attribute-based broadcast encryption scheme (CP-ABBE) supporting negative attributes and able to handle access policies in conjunctive normal form (CNF). The ciphertext length remains linear in the size of the access policy and in the number of revoked receivers but independent of all the users in the system. But this ciphertext is associated with the revoked users; when the revoked users change, the ciphertext needs to be reencrypted. Liu et al. in 2018 [20] proposed a solution adopting techniques on secure outsourcing of pairings to support outsourcing computation and adopting some techniques based on the tree-based scheme to solve user revocation and attribute revocation. But this scheme needs to reencrypt the ciphertext. Shangping et al. in 2018 [21] proposed an attribute revocation by delegating the update of the secret key and ciphertext. Premkamal et al. in 2019 [22] proposed a scheme of verifiable outsourced CP-ABE and transfer of the computational overhead to the proxy server. Li et al. in 2020 [23] proposed verifiable outsourced decryption scheme which can simultaneously verify the correctness for transformed ciphertext for the authorized users and unauthorized users. Gao et al. in 2020 [24] proposed a new trustworthy secure ciphertext-policy and attribute hiding access control scheme based on blockchain to solve the problems of high trust-building cost, single point of failure, and privacy leakage. Li et al. in 2021 [25] proposed an efficient extended file hierarchy CP-ABE scheme (EFH-CP-ABE) to encrypt multiple files on the same access level.

The above schemes improve the security, computational efficiency, and verifiability of the CP-ABE algorithm. But all these schemes need to update the secret keys and ciphertext when CSP judges if the user's authority is revoked, which will spend a lot of time.

2.3. The Application of CP-ABE Algorithm. Taha et al. in 2019 [26] proposed a scheme of vehicles cluster based on CP-ABE, which assigns the encryption tasks through the vehicle information to save the computational cost. Laaboudi et al. in 2019 [27] proposed the scheme based on group encryption CP-ABE applied to communication protection of IPS and IDS. Zhang et al. in 2021 [28] designed an online privacy-protective decryption testing algorithm using a partially HP-CP-ABE (PHP-CP-ABE) method to prevent the attribute values guessing attacks (AVGA). Sowjanya et al. in 2021 [29] proposed a lightweight key management mechanism for the CP-ABE scheme using Elliptic Curve Cryptography (ECC) to solve the problems of security, privacy, and computational complexity in IoT in medical care.

The above literature focus on application of CP-ABE in Vehicular Ad Hoc Networks, IoT network, and so forth. The focus of such applications takes attentions on security, efficiency, and privacy, especially in computational cost. But all these schemes need to update the secret keys and ciphertext when CSP judges if the user's authority is revoked, which will spend a lot of time.

3. Preliminaries

3.1. Symbols Abbreviations. Table 1 shows the symbols abbreviations and their corresponding descriptions.

3.2. Bilinear Maps. Define G and GT as two cyclic groups of prime order p. Then the Discrete Logarithm Problem on G and GT is a hard problem. A bilinear function map e is defined as $G * G -> \mathrm{GT}$; its properties are as follows [2]:

(1) $\forall g \in G$ and $a, b \in Z_P^*$, $e(g^a, g^b) = e(g, g)^{ab}$

Table 1: Symbols abbreviations.

Abbreviation	Description
T	The system tree
$T1$	The access control policy tree
$T2$	The tree identical with $T1$
SK	The secret keys set for the system tree T
$CT1$	Ciphertext encrypted by the user itself according to $S1$ with the elliptic curves parameter $G_{01}, g1$
$CT2$	Ciphertext encrypted by the CSP according to $S2$ with the elliptic curves parameter $G_{02}, g2$
$G_{01}, g1$	The parameters of constructing $T1$
$G_{02}, g2$	The parameters of constructing T 2
S_i	The ith user's attributes set
$SK1_i$	The ith user's secret keys set of $T1_{\mathrm{i}}$
$SK2_i$	The ith user's secret keys set of $T2_{\mathrm{i}}$
p, g, α, β	The parameters related to the algorithm of CP-ABE
$\mathrm{CT1_i}$	Ciphertext encrypted by the ith user itself according to S_i
$\mathrm{CT2_i}$	Ciphertext encrypted by the ith user according to S_i
$S1$	The attributes set of $T1$
$S2$	The attributes set of $T2$
$SK1$	The secret key set of $T1$
$SK2$	The secret key set of $T2$
$PK1$	The public key of $T1$
$PK2$	The public key of $T2$
$MSK1$	The main secret key of $T1$
$MSK2$	The main secret key of $T2$
$T1_i$	The access control tree corresponding to the ith user
$T2_i$	The tree identical with $T1_{\mathrm{i}}$
I_i	The indexes set to the ith user's attributes set
$\widetilde{C}, C, C_{y1}, C'_{y1}$	The components involved in $CT1$
C_{y2}, C'_{y2}	The components involved in $CT2$
M	The plain text

(2) $e(g, g) \neq 1$

3.3. BDHE Assumption. The BDHE problem [6, 14] is to compute $e(g, g)^{a^{q+1}s} \in G_T$, in which g is a generator of group G and $\overrightarrow{y} = (g_1, \ldots, g_q, g_{q+1}, \ldots, g_{2q}, g^s)$. Let g^i denote g^{a^i}.

If $|P_r[\mathscr{B}(\overrightarrow{y}, g, T = e(g, g)^{a^{q+1}s}) = 0] - P_r[\mathscr{B}(\overrightarrow{y}, T = R) = 0]| \geq \varepsilon$, we will regard that there exists a polynomial-time algorithm $\mathscr{B}$ that has advantage $\epsilon(x)$ to solve the decisional BDHE problem.

3.4. Lagrange Coefficient [30]

$$\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x - j}{i - j},\ i \in Z_p \text{ and } S \text{ is a set of elements in } Z_p. \tag{1}$$

3.5. Waters' CP-ABE Scheme

Setup. The algorithm is used to initialize the system. It takes as input an implicit security parameter and outputs the public parameter PK and a master key MK.

Encrypt (**PK**, **M**, $\mathbb{A}$). The algorithm is used to encrypt the message. It takes as input PK, a message M, and an access structure $\mathbb{A}$. It encrypts M and generates a ciphertext CT. Only when the attributes set that the user owns satisfies $\mathbb{A}$ can the user decrypt the message.

Key Generation (**MK**, **S**). The algorithm is used to produce the private keys. It takes as input the master key MK and a set of attributes S and outputs the private key SK.

Decrypt (**PK**, **CT**, **SK**). The algorithm is used to decrypt the ciphertext. It takes as input the public parameter PK, a ciphertext CT, and a private key SK. If the set of attributes S satisfies the access structure $\mathbb{A}$, then it will decrypt the ciphertext CT and return the message M [31].

3.6. Access Structure. Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $A \subseteq 2_{\{P_1, P_2, \ldots, P_n\}}$ is monotone if, $\forall B, C$, if $B \in \Lambda$ and $B \subseteq C$ then $C \in \Lambda$. An access structure (respectively, monotone access structure) is a collection (resp. monotone collection) Λ of nonempty subsets of $\{P_1, P_2, \ldots, P_n\}$; that is, $\Lambda \subseteq 2_{\{P_1, P_2, \ldots, P_n\}} / \{\phi\}$. The sets in Λ are called the authorized sets, and the sets not in Λ are called unauthorized sets [32].

3.7. Access Trees. Every nonleaf node of the tree represents a threshold gate, described by its children and a threshold value. Define $\mathrm{n_x}$ as the children number of a node x and $\mathrm{k_x}$ is its threshold value ($0 < k_x < n_x$). If the threshold gate is an OR gate and then $k_x = 1$, while the threshold gate is an AND gate and then $k_x = n_x$. Each leaf node x of the tree represents an attribute in the attribute set S and the value of it is k_x.

4. The Definition of Our Scheme

4.1. Formal Definition of Our Scheme. Based on Waters' scheme of CP-ABE [31], we propose our scheme which includes six steps and is shown as follows:

Setup. It takes as input $G_{01}, G_{02}, g1$, and $g2$ and outputs the public keys PK1 and PK2 and the master keys MSK1 and MSK2, respectively, for trees T1 and T2. This step is executed by DO.

EncryptCT (**PK**, **M**, **T**)**.** It takes $PK1, M, T1$ as input and generates $CT1$. At the same time, it takes $PK2, T2$ as input and generates $CT2$.

KeyGen (**MSK**, **S**). It takes $MSK1$ and the user's attributes $S \subset A$ as input and generates the decryption secret key $SK1$ based on tree $T1$. At the same time, it takes $MSK2$ and $S \subset A$ as input and generates the decryption key $SK2$ based on tree $T2$.

Compute T ($\mathbf{C_x}, \mathbf{C_x^{'}}$, **SK**, **x**). It takes C_X, C_x' involved in CT and SK as input, in which x is the node of the tree. The algorithm is recursively executed from the leaf node of the tree until the root node R. If SK satisfies the condition, the result of the algorithm is F_R; otherwise, the result is $\perp$.

JudgeT (**CT**2, **SK**2, $\mathbf{I_i}, \mathbf{C_{y2}}, \mathbf{C_{y2}^{'}}$). CSP judges if the attributes set I_i satisfies $I_i \subseteq S$ by calling $ComputeT(C_x, C_x^{'}, SK, x)$. If the result is not $\perp$, it proves that $SK2$ can decrypt $CT2$; that is, $I_i \subseteq S$, which proves that $SK1$ can decrypt $CT1$.

Decrypt (**CT**1, $\mathbf{SK1_i}$). It takes $CT1, SK1$ as input and obtains the plaint text M.

4.2. Threat Model and Security Requirement

4.2.1. CPA Security. Waters' scheme [31] satisfies CPA security. Our scheme is constructed based on Waters'; so our scheme also satisfies CPA security. We design a game between the challenge and the adversary to describe the process which is as follows:

Setup. The challenger generates the master keys $MSK1$ and $MSK2$ and the public keys PK1 and PK2, respectively, for trees $T1_W$ and $T2_W$.

Query phase 1. The adversary constructs many attribute sets $S1, S2, S3 \ldots$ to issue corresponding requests of secret keys to the challenger. At the same time, the adversary sends the encrypted index sets corresponding to the attribute sets—$E(I1), E(I2), E(I3)$... to the challenger. The challenger generates $SK1, SK2, SK3 \ldots$. and adds $\langle j++, Si, SK1i, Ii \rangle$ to table T.

Challenge. The adversary sends two equal-length messages $M0, \rho_x^*, E(Ik)$ and $M1, \rho_x^*, E(Ik)$ to the challenger. In these messages, Si constructed in query phase does not satisfy ρ_x^*. The challenger randomly selects $\beta \in (0, 1)$ and encrypts M_β as CT^* with the access structure ρ_x^* and then sends CT^* to the adversary.

Query phase 2. Run query phase 1. But, in this phase, the attribute sets issued by the adversary cannot be equal to the access structure ρ_x^*.

Guess. The adversary outputs $\beta' \in \{0, 1\}$ to the challenger.

The advantage of the adversary in this game is defined as $|\Pr[\beta - \beta'] - (1/2)|$ where the probability is taken over the random bits used by the challenger and the adversary.

4.2.2. Data Confidentiality. CSP has no right to access the encrypted data and the user who does not satisfy the access control policy cannot visit the ciphertext.

4.2.3. Collusion Resistance. Even if multiple users combine their attribute keys, they cannot decrypt the ciphertext.

(i) *Backward Security*. Backward security means that when a new user joins, if he/she is not authorized, he/she cannot access the plain text.

4.2.4. Forward Security. Forward security means that if a user is revoked, even if he/she preserved the secret key before being revoked, he/she has no right to access the plain text.

5. The Detailed Implementation Scheme

We construct an identical tree $T2$ that has the completely same structure as the access control tree $T1$. The access control tree $T1$ is shown in Figure 2(b) marked with a yellow rectangle and the identical tree $T2$ is shown in Figure 2(c) marked with a red rectangle. They have the completely same structure including nodes and the relationship among nodes.

5.1. Architecture of Our Scheme. The CP-ABE system consists of DO (Data Owner), CSPs (Cloud Service Providers), DU (Data User), and AA (Attribute Authority). The architecture of our system is shown in Figure 3 and the work flow of it is as follows:

(1) DO calls **Setup** to compute MSK1, PK1 and MSK2, PK2 and then publishes PK1 and PK2.

(2) DO calls **EncryptCT** to encrypt M as CT1.

(3) DO calls **EncryptCT** to compute CT2.

(4) AA generates a set of private keys $SK1_i$ based on tree $T1$ according to his/her attributes S_i for each user, in which i denotes the ith user, and a set of secret keys $SK2 = \{ski | i = 1, 2...\}$ according to the identical tree $T2$ shared by all the users.

(5) AA transmits SK2 to DO.

(6) DO uploads {CT1, CT2, SK2} to CSP.

(7) The ith DU requests to visit CT1 to CSPs.

(8) DU logs on AA and asks for private keys.

(9) AA sends $SK1_i$ to DU.

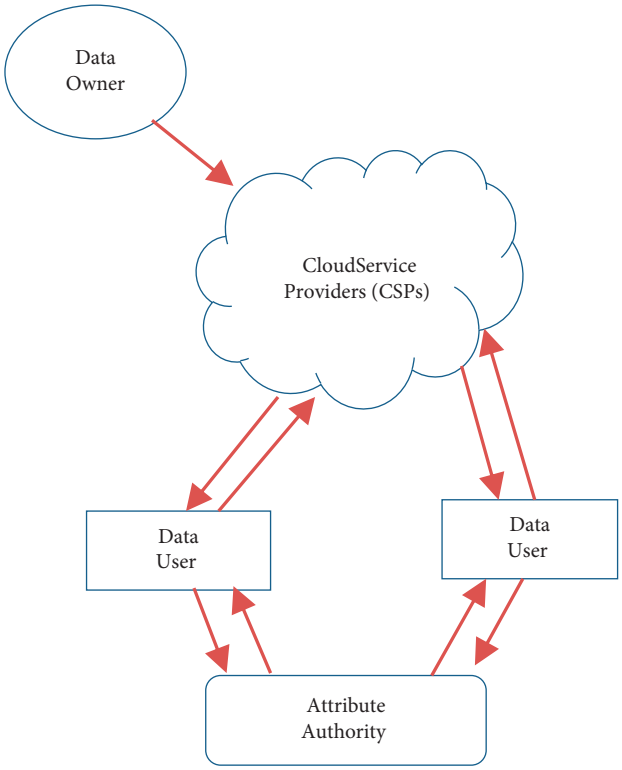

Figure 3: The architecture of our scheme.

(10) AA encrypts the indexes set I_i which is corresponding to the attributes set of $SK1_i$ with the public key of CSPs and sends it to DU:

$$E_{ui}\left(SK1_i, E_{csp}\left(I_i\right)\right), I_i = \{i | i = \text{index}\left(\text{att}\left(x\right)\right) = \text{index}\left(SK1_i\right), x \in S_i\}. \tag{2}$$

Now we take Figure 2 as an example to illustrate the relationship between I_i and S_i. Suppose that the attributes set S_i is $\{A1, A3, A4\}$; then the indexes set I_i is $\{1, 3, 4\}$.

(11) DU sends $\left\{SK1_i, E_{csp}\left(I_i\right)\right\}$ to CSPs.

(12) CSP reconstructs $SK2_i$ from SK2 according to I_i; for example, if $I_i = \{1, 3, 4\}$, $SK2_i = \{sk1, sk3, sk4\}$.

(13) CSP calls JudgeT $(CT2)$ to decide if the user has the right to access CT1. If the returned result is equal to $\perp$, CSP rejects the DU's request.

(14) If the returned result is not equal to $\perp$, CSP sends CT1 to DU.

(15) DU calls DecryptCT to decrypt CT1.

The algorithm process is given in Algorithm 1.

5.2. The Detailed Realization. Our scheme is based on the scheme in [31]. The basic principles of the algorithm Setup, EncryptCT, KeyGen, and DecryptCT are completely the same as those of the scheme in [31]. The difference between the scheme in [31] and ours is that we introduce the identical tree $T2$ and construct $CT2$ and another new algorithm JudgeT, which helps to decide whether to allow $CT1$ to be accessed. Before defining the algorithms, we will illustrate some key notations:

(1) $T1_W$ and $T2_W$ denote the system tree when a system is fixed and they have the same structure as the system tree

(2) $T1_i$ and $T2_i$ represent the trees corresponding to the ith user

The detailed algorithms are as follows:

Setup. DO calls Setup to, respectively, generate MSK1, PK1 and MSK2, PK2 for system trees $T1_W$ and $T2_W$. The detailed function of **Setup** algorithm is that it chooses a bilinear group G_0 of prime order p with generator g. Next it will choose two random exponents $\alpha, \beta \varepsilon Z_P$. The public key is published as

$$PK = G_0, g, h = g^{\beta}, e\left(g, g\right)^{\alpha}. \tag{3}$$

The master key MSK is (β, g^{α}). So

$$\begin{aligned} PK1 &= G_{01}, g1, h1 = g^{\beta 1}, e\left(g1, g1\right)^{\alpha 1}, \\ MSK1 &= \left(\beta 1, g2^{\alpha 1}\right), \\ PK2 &= G_{02}, g2, h2 = g^{\beta 2}, e\left(g2, g2\right)^{\alpha 2}, \\ MSK2 &= \beta 2, g2^{\alpha 2}. \end{aligned} \tag{4}$$

EncryptCT (**PK**, **M**, **T**). The encryption algorithm encrypts a message M according to the access policy. The algorithm first chooses a polynomial q_x for each node x (including the leaves) in the tree. These polynomials are chosen in the following way in a top-down manner, starting from the root node R. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node; that is, $d_x = k_x - 1$.

Starting with the root node R, the algorithm chooses a random $s \in Z_p$ and sets $q_R(0) = s$. Then, it chooses d_R other points of the polynomial q_R randomly to define it completely. For any other node x, it sets $q_x(0) = q_{\text{parent}(x)}(\text{index}(x))$ and chooses d_x other points randomly to completely define q_x.

Let Y1 be the set of leaf nodes on tree $T1$ and let Y2 be the set of leaf nodes on tree $T2$. We can obtain $CT1$ and $CT2$.

$$\begin{aligned} CT1 &= \begin{pmatrix} T1, \widetilde{C} = Me\left(g1, g1\right)^{\alpha 1 s}, C = h1^{s}, \\ \forall y \in Y1: C_{y1} = g1^{q1_y(0)}, C'_{y1} = H\left(\text{att}\left(y1\right)\right)^{q1_y(0)} \end{pmatrix}, \\ CT2 &= \left(T2, \forall y \in Y2: C_{y2} = g2^{q2_y(0)}, C'_{y2} = H\left(\text{att}\left(y2\right)\right)^{q2_y(0)}\right). \end{aligned} \tag{5}$$

KeyGen (**MSK**, **S**). For $T1$, the key generation algorithm will take as input a set of attributes $S_i \subseteq Y1 = A$ which is possessed by a user and authorized to the user by AA and output a key that identifies with that set. The

DO:
Setup () $\longrightarrow$ MSK1, PK1 and MSK 2, PK2
EncryptCT (PK1, M, T1) $\longrightarrow$ CT1
EncryptCT (PK2, T2) $\longrightarrow$ CT2
AA:
KeyGen $(\mathbf{MSK1}, \mathbf{S_i}) \longrightarrow SK1_i$
KeyGen $(\mathbf{MSK2}, \mathbf{T2}) \longrightarrow SK2$
Sends SK2 to DO
DO:
Uploads {CT1, CT2, SK2} to CSP
DU(the ith user):
Request to visit CT1 to CSP
Log on AA to ask for private key
AA:
Sends $SK1_i$ to DU
Compute:
$I_i = \{i | i = \text{index}(\text{att}(x)) = \text{index}(SK1_i), x \in S_i\}$
$E_{ui}(SK1_i, E_{\text{csp}}(I_i))$
Send to DU
DU:
Send $\left\{\text{SK1}_i, E_{\text{csp}}(\text{I}_i)\right\}$
CSP:
KeyGen $(\mathbf{MSK2}, \text{I}_i) \longrightarrow SK2_i$
JudgeT $(CT2, SK2_i, I_i, C_{y2}, C_{y2}') \longrightarrow$ x
If x == $\perp$, return reject
else
Decrypt $(\mathbf{CT1}, \mathbf{SK1_i}) \longrightarrow \mathbf{M}$
Send M to DU

Algorithm 1: An improved CP-ABE algorithm based identical tree.

algorithm first chooses a random number $r \in Z_p$ and then random $r_j \in Z_p$ for each attribute $j \in S$. Then it computes the key for user u_i as follows:

$$SK1_i = \left(D = g1^{(\alpha 1 + r/\beta 1)}, \quad \forall j \in S_i \subseteq A: D_j = g1^r \times H(j)^{r_j}, D_j' = g1^{r_j}\right). \tag{6}$$

For T2, the objective of introducing it is to judge if the user has right to access ciphertext $CT1$, so all the users share the secret keys $SK2 = \{sk2_i | i = 1, 2, \ldots, \text{num}(Y2)\}$, in which num denotes the number of the leaf nodes. AA computes every key $sk2_i$ corresponding to every leaf node y on tree T2; that is, AA generates a key for every attribute y ∈ Y2 = B. Because these attributes are shared by every user, the keys are only generated once and there is no difference for every user. So, in the algorithm of generating $SK2$, we do not introduce random r to distinguish every user like the process of generating $SK1_i$. The formula is as follows:

$$SK2 = \left(D = g2^{(\alpha 2/\beta 2)}, \quad \forall k \in B: D_k = H(j)^{r_k}, D_k' = g2^{r_k}\right). \tag{7}$$

The private key SK2 only is generated once and uploaded to CSPs for all the users to share.

JudgeT $(\mathbf{CT2}, \mathbf{SK2}, \mathbf{I_i}, \mathbf{C_{y2}}, \mathbf{C_{y2}'})$. After receiving I_i from the *ith* DU, CSPs will obtain the indexes in set I_i. For every $j \in I_i$, CSP finds the corresponding key $sk2_j \in SK2$ and then forms a key set $SK2_i$.

This algorithm judges if $SK2_i$ satisfies the access policies of tree $T2_i$ by calling Compute$T(C_{y2}, C_{y2}', SK2_i, R_2)$, in which $R2$ is the root node of $T2_i$. If the result F_{R2} of Compute$T(C_{y2}, C_{y2}', SK2_i, R_2)$ is a certain value, we think that $SK2_i$ satisfies the access policies. So, we can conclude that CT1 can be sent to the user u_i. Otherwise, we reject the request of the user. The computing process of Compute$T(C_x, C_x', SK, x)$ is as follows:

ComputeT $(\mathbf{C_x}, \mathbf{C_x'}, \mathbf{SK}, \mathbf{x})$. First we define a recursive function Compute$T(C_x, C_x', SK, x)$ that takes as input C_x and C_x' involved in ciphertext CT; SK is the secret key associated with attributes set S authorized by AA to DU and x is a node of tree T. Define n_x as the children

number of a node x and k_x is its threshold value ($0 < k_x < n_x$). If the threshold gate is an OR gate, then $k_x = 1$, and if the threshold gate is an AND gate, then $k_x = n_x$. Each leaf node x of the tree represents an attribute in the attribute set S and the value of it is k_x.

Now we are going to compute the function $\mathrm{Compute}T(C_x, C_x', SK, x)$, which is proposed in the scheme in [31]. If x is the leaf node of the tree and $i = \mathrm{att}(x) \in S$, then

$$F_x = \mathrm{Compute}\,T\left(C_x, C_x', SK, x\right) = \frac{e\left(D_i, C_x\right)}{e\left(D_i', C_x'\right)} = \frac{e\left(g^r \times H(i)^{r_i}, g^{q_x(0)}\right)}{e\left(g^{r_i}, H(i)^{q_x(0)}\right)} = e(g,g)^{rq_x(0)}. \tag{8}$$

If $i \notin S$, then $\mathrm{Compute}T(C_x, C_x', SK, x) = \perp$.

When x is a nonleaf node, the computing process is as follows:

Let the children of x be $z_i, i \in (1, \ldots, n_x)$ and let S_x be the attributes set on the threshold gate. The algorithm calls $\mathrm{Compute}T(C_x, C_x', SK, z_i)$. Only when $k_x < |S_x| < n_x$ can F_{z_i} return a certain value. Otherwise, $F_{z_i} = \perp$. The computing process of F_x is as follows:

$$\begin{aligned} F_x &= \prod_{z_i \in S_x} F_{z_i}^{\Delta_{i,S_x'}(0)} = \prod_{z_i \in S_x} \left(e(g,g)^{r \cdot q_{z_i}(0)}\right)^{\Delta_{i,S_x'}(0)}, \\ &= \prod_{z_i \in S_x} \left(e(g,g)^{r \cdot q_x(i)}\right)^{\Delta_{i,S_x'}(0)} = e(g,g)^{r \cdot q_x(0)}. \end{aligned} \tag{9}$$

$\mathrm{Judge}T(CT2, SK2_i, I_i, C_{y2}, C_{y2}')$ recursively calls $\mathrm{Compute}T(C_{y2}, C_{y2}', SK2_i, x)$ until x is equal to R2 and F_{R2} is obtained, in which R_2 is the root node of tree $T2_i$. If $F_{R2} = \perp$, it means that the attributes set of the user does not satisfy the access control structure. CSP rejects to provide CT1 to the user.

Decrypt $(\mathbf{CT1}, \mathbf{SK1_i})$. If F_{R2} returns a certain result, it means that the attributes set of the user satisfies the access control structure. CSP will send CT1 to the user. The user will recursively call $\mathrm{Compute}T(C_{y1}, C_{y1}', SK1_i, x)$ until finally obtaining $F_{R1} = e(g,g)^{r \cdot q_{R1}(0)} = e(g,g)^{r \cdot s}$. At last, the plain text is decrypted and got by computing

$$\frac{\widetilde{C}}{(e(C,D))/F_{R1}} = \frac{\widetilde{C}}{\left(e\left(h^s, g^{(\alpha+r/\beta)}\right)/e(g,g)^{rs}\right)} = M. \tag{10}$$

6. Security Analysis

6.1. Security Proof

Theorem 1. *Suppose that the scheme of Waters [31] is a CPA-secure CP-ABE. Then our scheme is also CPA-secure.*

Proof. Suppose that there exist a polynomial-time adversary $\mathscr{A}$ that can attack our scheme with advantage ϵ in the CPA-secure model, a challenger $\mathscr{C}$ that is a simulator to the scheme of Waters, and a simulator of our scheme $\mathscr{B}$ that can attack Waters' scheme with advantage ϵ in the CPA-secure model:

Setup. $\mathscr{C}$ generates $MSK1$ and $PK1 = G_{01}, g1, h1 = g^{\beta 1}, e(g1, g1)^{\alpha 1}$ according to Waters' scheme. Then C sends PK1 to $\mathscr{B}$. $\mathscr{B}$ generates $PK2 = G_{02}, g2, h2 = g^{\beta 2}, e(g2, g2)^{\alpha 2}$ and MSK2 according to our scheme and publishes PK1 and PK2.

Query phase 1. $\mathscr{A}$ constructs many attributes set $S1, S2, S3 \ldots$ to issue corresponding requests of secret keys to $\mathscr{B}$. At the same time, $\mathscr{A}$ sends $E(I1), E(I2), E(I3) \ldots$ to $\mathscr{B}$. $\mathscr{B}$ sends Si to $\mathscr{C}$. $\mathscr{C}$ generates $SK1, SK2, SK3 \ldots$ to $\mathscr{B}$. Then B sends them to $\mathscr{A}$. $\mathscr{B}$ generates a Table T and adds $\langle j++, Si, SK1i, Ii \rangle$ to T.

Encrypt M^*. $\mathscr{A}$ constructs $\langle M^*, \rho^*, E(IK^*) \rangle$ to $\mathscr{B}$ and $\mathscr{B}$ sends it to $\mathscr{C}$ for encryption. $\mathscr{C}$ randomly selects $s \in Z_P$ and runs encryption algorithm $\mathrm{Encrypt}CT$ and returns $CT1^*$ to $\mathscr{B}$. *B runs encryption algorithm and returns* $CT2^*$. *Then* $\mathscr{B}$ *searches table* T *for* $I\mathrm{k} == Ii$.

(a) *If the query result is not null,* $\mathscr{B}$ *calls* $\mathrm{Judge}T$. *If the return value of* $\mathrm{Judge}T$ *is certain,* $\mathscr{B}$ *sends* $CT1^*$ *to* $\mathscr{A}$. *If the return value is* $\perp$, $\perp$ *is returned to* $\mathscr{A}$.

(b) *If the query result is null,* $\mathscr{B}$ *returns* $\perp$ *to* $\mathscr{A}$.

Challenge. $\mathscr{A}$ sends two equal-length messages $\mathrm{M0}, \rho_x^*, E(Ik)$ and $\mathrm{M1}, \rho_x^*, E(Ik)$ to $\mathscr{B}$ and $\mathscr{B}$ sends them to $\mathscr{C}$. In these messages, Si constructed in query phase does not satisfy ρ_x^*, so IK^* is not equal to IK. $\mathscr{C}$ randomly selects $\beta \in (0,1)$ and encrypts M_β as CT^* by calling the step of Encrypt M^*.

Query phase 2. Run Query phase 1, but Si constructed in query phase 1 does not satisfy ρ_x^*.

If $\mathscr{A}$ receives $\perp$, $\mathscr{A}$ generates a random number $\beta' \in \{0,1\}$. Then, in this situation, $\epsilon = \Pr(\beta = \beta') - (1/2) = (1/2) - (1/2) = 0$. If what $\mathscr{A}$ receives is not $\perp$, according to the scheme of Waters, ϵ can be negligible.

Guess. $\mathscr{A}$ outputs $\beta' \in \{0,1\}$ to $\mathscr{C}$.

Because the algorithms of Setup, EncryptCT, $KeyGen, Decrypt$ $KeyGen, Decrypt$ are all identical to Waters' algorithms and Waters' scheme satisfies CPA security, our scheme also satisfies CPA security.

The proof of Theorem 1 is complete. □

6.2. Resisting Collusion Attacks. Our scheme employs the method of the scheme in [31] to encrypt the plain text and generate the secret key for every user. The technique

Table 2: Comparisons of functionalities with the previous works.

	(1)	(2)	(3)
Scheme in [31]	Yes	Yes	Yes
Scheme in [13]	No	Yes	Yes
Scheme in [21]	No	Yes	Yes
Our scheme	No	No	No

Table 3: The meanings of notations.

Notation	Meaning
p	The pairing of operation
e	The group exponentiation
$\|Y\|$	The number of attributes in the system
$\|S\|$	The average number of attributes possessed by one user
$\|K\|$	The number of revoked users
$\|\widetilde{S}\|$	The average number of revoked attributes
$\|I\|$	The number of attributes for decryption

generates a random number r for every user when distributing secret key, which means that only the secret keys with same r can decrypt CT, while every user has different r. So, even if multiple users collude and every user provides a secret key and at last they collect every secret key corresponding to every attribute, the collusion is not successful for parameter r corresponding to every user is different, which makes it is impossible to conspire among many users. The security of the scheme has been proved in literature [31].

6.3. Invisible to CSPs. Although trees T2 and T1 have the same structure, the private keys of T2 cannot decrypt CT1 for the master key and public parameters are not the same as trees T1 and T2. So, although CSP can compute the value of F_{R2}, it cannot compute the value of F_{R1}. Therefore, CSP cannot decrypt ciphertext CT1. So, the message is safe to be preserved on CSP.

6.4. Backward Safe. Our scheme is backward safe. When a new user joins the system, if he/she did not distribute the private key corresponding to the access control tree of the old ciphertext, he/she cannot decrypt the old CT. So, our scheme is backward safe.

6.5. Forward Safe. Suppose that after some attributes of one user are revoked by AA, he/she does not satisfy the access control policy; then he/she cannot access the ciphertext. If he/she can decrypt the ciphertext before the attributes are revoked and he/she preserved the secret key, for he/she is rejected to visit the ciphertext if he/she does not satisfy the access control policy at the current, he/she cannot obtain the ciphertext CT. So, even if the secret key before the attributes are revoked is preserved by the user, he/she cannot get the CT and decrypt it. So, our scheme is forward safe.

6.6. Protection of the Attribute Privacy and Access Control Policy. Although tree T2 has the same structure as tree T1, we cannot discover the attribute names and relationship from CT1. So, CSPs cannot discover the attribute names and relationship from CT2 although CSPs have abilities to decrypt tree T2. Therefore, our scheme will not leak the privacy of the attributes and relationship.

6.7. Revocation Analysis. Our scheme constructs tree $T2$ which is identical with the real access control tree $T1$ for every user. The two trees have the same structure. The secret keys corresponding to the leaf nodes of tree $T2$ are shared by all the users and preserved on CSP. CSP judges if the user has the access to tree $T1$ by the secret indexes transmitted from AA. CSP finds the real secret keys according to the secret indexes to compute the value of F_{R2} of tree $T2$. If the returned result is null, it proves that the user has no right to access the ciphertext and CSP rejects to send $CT1$ to the user. So, although the user saved the secret keys when he had had right to visit $CT1$, he/she cannot visit the message for this time he/she has been revoked some attributes which results in not obtaining $CT1$. So, our revocation scheme is safe.

CSP judges if the user has the right to visit the CT according to the secret indexes which represent the attributes that the user possesses now. If the indexes set I is null, it means that the user has no attribute authorized by AA, that is, the user has been revoked. So, our scheme can realize the user's attribute revocation and user revocation.

7. Analysis of Performance

7.1. Comparisons with the Previous Schemes in Function. In this section, we will compare our scheme with the previous schemes in revocation function from three aspects: (1) if the secret keys are updated; (2) if the ciphertext is updated; (3) if the secret key or ciphertext is related to the number of revoked users or attributes. They are shown in Table 2.

From Table 2, we can see that only our scheme updates neither the secret keys nor the ciphertext. So, our scheme has no overhead on the secret keys and ciphertext update. While in other schemes not only are the secret keys or ciphertext updated, but also the overhead of the update is related to the number of the users or the attributes. Once the number of

Table 4: Comparison of computation complexity.

	Setup	KeyGen	Encrypt	Decrypt	Revocation	Total
Scheme in [31]	$1 * e + 1 * p$	$(2 * \|S\| + 1) * e * U$	$(2 * \|Y\| + 1) * e + 1 * p$	$\|I\| * p$	$(2 * \|\widetilde{S}\| + 2) * e * \|K\|$	$O(\|I\|) * p + O(\|S\|\|U\| + \|Y\| + \|\widetilde{S}\|\|K\|) * e$
Scheme in [13]	$p + 2 * e$	$(3 * \|S\| + 2) * e * U$	$(2 * \|Y\| + 2) * e + 1 * p$	$2 * (\|I\| + 1) * p + (\|I\| + \log\|Y\|) * e$	$((\|Y\| + 3) * e + \|\widetilde{S}\| * e *)\|K\|$	$O(\|Y\|) * p + O(\|S\|\|U\| + \|I\| + \|Y\|\|K\| + \|\widetilde{S}\|\|K\| + \log\|Y\|) * e$
Scheme in [21]	$p + (\|Y\| + 3) * e$	$(\|Y\| + 4) * e * U$	$(3 * \|Y\| + 2) * e + 1 * p$	$(2 * \|I\| + 2) * p + 2 * \|I\| * e$	$2 * \|\widetilde{S}\| * e * \|K\|$	$O(\|I\|) * p + O(\|Y\|\|U\| + \|\widetilde{S}\|\|K\| + \|Y\|) * e$
Our scheme	$(2 * e + 1 * p) + p1$	$(2 * \|S\| + 1) * e * U + (2 * \|Y\| + 1) * e$	$2 * ((2 * \|Y\| + 1) * e) + 1 * p + 1 * p1$	$\|I\| * p + \|I\| * p1$	0	$O(\|I\|) * p + O(\|S\|\|U\| + \|Y\|) * e$

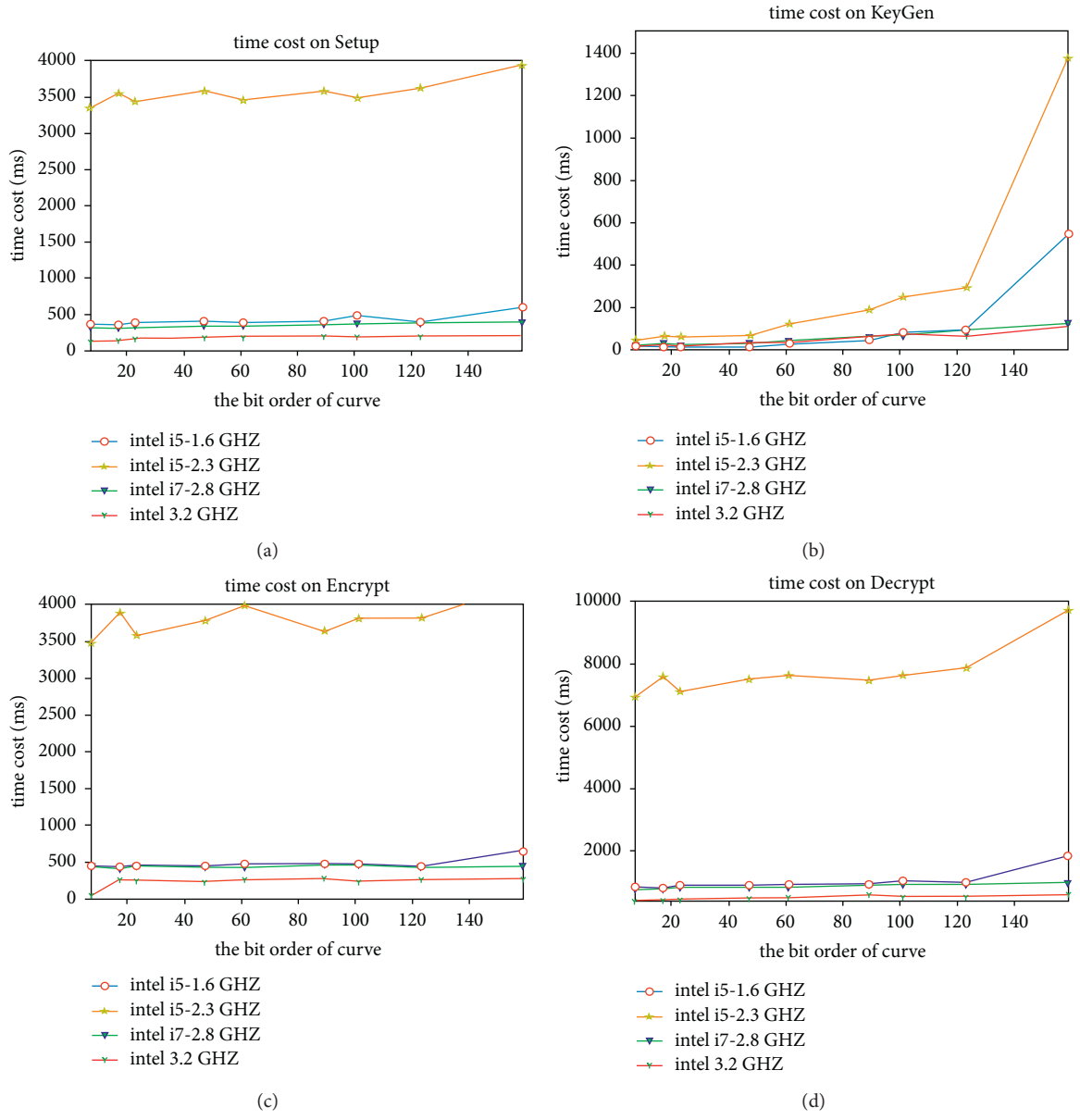

Figure 4: (a) Setup. (b) KeyGen. (c) Encrypt. (d) Decrypt.

the users or attributes becomes very large, the computation time or storage overload will increase rapidly.

7.2. Comparisons with the Previous Schemes in Computation Complexity. Table 3 shows the meanings of the notations which appear in Table 4. Table 4 lists the comparison on computation complexity of four schemes. From the column of **TOTAL**, we can see that our scheme consumes the shortest time compared to three other schemes in computation complexity.

8. Optimization

In this scheme, we construct an identical tree T2 to judge if the user who possesses the private keys can decrypt the cipher text CT1 by decrypting CT2. Only can CSPs compute a certain value, it can prove that the user has the ability to decrypt the cipher text. So, it is only related to the ability of decrypting CT2 and has nothing to do with the time complexity of computing F_R. In addition, every time before the user visits CT1, CSPs will first call the function **JudgeT** to compute and decrypt CT2. So, the time consumed on decrypting CT2 influences the system efficiency. Therefore, we can adjust the initialization parameters corresponding to tree T2 for decreasing the time consumed in decrypting CT2. In our system, the initialization parameter is prime order p in Z_p. From the experiments in Section 8, we can discover that the performance of the whole system is improved a lot by adjusting parameter p with the same structure as the real access control tree T1. The function of tree T2 is to help judge whether the user has the authorization to access ciphertext CT1.

9. Experiments

The experimental environments are four laptop computers: the first with Windows 10 system with Intel® Core™ i5 CPU @ 1.60 GHz and 8.00 GB RAM, the second with Windows 10 system with Intel® Core™ i5-6300HQ CPU @ 2.30 GHz and 12.00 GB RAM, the third with Windows 10 system with Intel i7-7700 CPU@2.8 GHz and 8 GB memory, and the fourth with Windows 10 with Intel CPU @ with 3.2 GHz and 2 GB memory. The y-axis in Figure 4 shows the time cost in the four phases, respectively, in four computers and the x-axis represents the different bit order of p in the curve group. From the four figures, we can see the following: (1) the cost time becomes bigger with the increase of the bit order; (2) the cost times in the four phases at the same bit order are different on different CPU and the cost time on CPU 3.2 GHz at the same bit orbit is the minimum and that on CPU i5 1.6 GHz is the maximum, which means that the computing ability of CPU influences the cost time; (3) the cost time in the maximum bit order (159) is about 10 times that in the minimum bit order (7); (4) the decryption time which is the phase's most consuming time on CPU 3.2 GHz in 7-bit order is about 25 times that on CPU i5 1.6 GHz in 159-bit order. According to the optimization theory in Section 7, the introduction of tree T2 is just to judge whether the user has the right to decrypt CT1 and it is not constrained by confidentiality. We want it to compute fast. From the above experiments, we can see that the small bit order and the CPU with the highest performance can raise the computing speed. Therefore, we can choose the 7-bit prime order and 3.2 GHz CPU as the server of CSP to decrypt tree T2. For ciphertext CT1 must satisfy confidentiality and does not tend to be decrypted, in this situation, the big bit order is required. In addition, the client device is generally in the low performance. So, we choose 159-bit order and CPU i5 1.6 GHz to simulate the client. At this time, we can discover from Figure 4(d) that the cost time in CSP is just 406 ms and about 25 times that in the client, which means that the time cost consumed in decrypting CT2 can be ignored. So, our scheme realizes the attribute revocation and does not add too much computation overload on the system.

10. Conclusion

In this paper, we innovatively propose a scheme of user's attribute revocation that is to construct an identical tree T2, preserved on CSPs, according to the access control tree T1 and judge whether DU has the right to access ciphertext CT1 by decrypting CT2. Finally, we analyze the security and time efficiency of our scheme and prove that our scheme is viable.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the youth fund of Shanxi University of Finance and Economics (Grant no. QN2015014), Shanxi Soft Science Research Project (Grant no. 2018041039-2), teaching reform project of Shanxi University of Finance and Economics (Grant no. 2018226), and research project of Shanxi Statistical Society (Grant no. KY[2019]164).

References

[1] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," *Advances in Cryptology CRYPTO 2005*, Springer, Berlin, Germany, 2005.

[2] A. Lewko, A. Sahai, and B. Waters, "Revocation systems with very small private keys," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, pp. 273–285, Oakland, CA, USA, May 2010.

[3] J. Ye, W. Zhang, and S. L. Wu, "Attribute-based fine-grained access control with user revocation," *Information and Communication Technology*, vol. 8407, 2014.

[4] X. Lian, X. Li, and R. Lu, "An efficient and secure user revocation scheme in mobile social networks," in *Proceedings of the 2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*, Houston, TX, USA, December 2012.

[5] H. Cui, R. Deng, and X. Ding, *Attribute-Based Encryption with Granular Revocatio*, Singapore Management University, Singapore, 2016.

[6] C.-I. Fan, V. S.-M. Huang, and H.-M. Ruan, "Arbitrary-state attribute-based encryption with dynamic membership," *IEEE Transactions on Computers*, vol. 63, no. 8, pp. 1951–1961, 2014.

[7] J. De and S. Ruj, "Efficient decentralized attribute based access control for mobile clouds," *IEEE Transactions on Cloud Computing*, vol. 1, 2017.

[8] Z. L. Jiang, R. Zhang, and Z. Liu, *A Revocable Outsourcing Attribute-Based Encryption Scheme*, Springer, Berlin, Germany, 2016.

[9] R. Yadav, W. Zhang, and K. Li, "Managing overloaded hosts for energy-efficiency in cloud data centers," *Cluster Computing*, pp. 1–15, 2021.

[10] R. Yadav, W. Zhang, and K. Li, "An adaptive heuristic for managing energy consumption and overloaded hosts in a cloud data center," *Wireless Networks*, vol. 26, 2018.

[11] S. Joyee De and S. Ruj, "Decentralized access control on data in the cloud with fast encryption and outsourced decryption," in *Proceedings of the GLOBECOM 2015-2015 IEEE Global Communications Conference*, IEEE, 2015.

[12] S. Yu, C. Wang, and K. Ren, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security-ASIACCS*, p. 261, Beijing, China, April 2010.

[13] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1214–1221, 2011.

[14] X. Xie, H. Ma, and J. Li, "New ciphertext-policy attribute-based access control with efficient revocation," in *Proceedings of the 2013 International Conference on Information and Communication Technology*, Springer, Cape Town, South Africa, December 2013.

[15] T. Naruse, M. Mohri, and Y. Shiraishi, "Provably secure attribute-based encryption with attribute revocation and grant function using proxy re-encryption and attribute key for updating," *Human-centric Computing and Information Sciences*, vol. 5, no. 1, p. 8, 2015.

[16] Y. Cheng, Z.-Y. Wang, J. Ma, J.-J. Wu, S.-z. Mei, and J.-C. Ren, "Efficient revocation in ciphertext-policy attribute-based encryption based cryptographic cloud storage," *Journal of Zhejiang University-Science C*, vol. 14, no. 2, pp. 85–97, 2013.

[17] Genlang, Chen, and Zhiqian, "Generic user revocation systems for attribute-based encryption in cloud storage," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, 2018.

[18] S. S. Tu, S. Z. Niu, and H. Li, "Fine-grained access control and revocation for sharing data on clouds," in *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, Shanghai, China, May 2012.

[19] B. Wesolowski and J. Pascal, "Ciphertext-policy attribute-based broadcast encryption with small keys," in *Proceedings of the International Conference on Information Security and Cryptology*, Springer International Publishing, Seoul, South Korea, November 2016.

[20] Z. Liu, W. Xuan, and C. Lei, "White-box traceable dynamic attribute based encryption," in *Proceedings of the 2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, Shenzhen, China, December 2018.

[21] W. Shangping, Z. Duo, and Z. Yaling, "Efficiently revocable and searchable attribute-based encryption scheme for mobile cloud storage," *IEEE Access*, vol. 1, 2018.

[22] P. K. Premkamal, S. K. Pasupuleti, and P. Alphonse, "A new verifiable outsourced ciphertext-policy attribute based encryption for big data privacy and access control in cloud," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 2, 2019.

[23] J. Li, Y. Wang, and Y. Zhang, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Transactions on Services Computing*, vol. 1, 2020.

[24] S. Gao, G. Piao, and J. Zhu, "TrustAccess: a trustworthy secure ciphertext-policy and attribute hiding access control scheme based on blockchain," *IEEE Transactions on Vehicular Technology*, vol. 99, p. 1, 2020.

[25] J. Li, N. Chen, and Y. Zhang, "Extended file hierarchy access control scheme with attribute based encryption in cloud computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, 2019.

[26] A. Mbt, A. Ct, and A. Os, "A cluster of CP-ABE microservices for VANET," *Procedia Computer Science*, vol. 155, pp. 441–448, 2019.

[27] Y. Laaboudi, A. Olivereau, and N. Oualha, "An Intrusion Detection and Response Scheme for CP-ABE-Encrypted IoT Networks," in *Proceedings of the 2019 10th IFIP International Conference on New Technologies, Mobility and Security*, NTMS, Canary Islands, June 2019.

[28] Z. Zhang, W. Zhang, and Z. Qin, "A partially hidden policy CP-ABE scheme against attribute values guessing attacks with online privacy-protective decryption testing in IoT assisted cloud computing," *Future Generation Computer Systems*, vol. 123, pp. 181–195, 2021.

[29] K. Sowjanya, M. Dasgupta, and S. Ray, "A lightweight key management scheme for key-escrow-free ECC-based CP-ABE for IoT healthcare systems," *Journal of Systems Architecture*, vol. 117, no. 2, Article ID 102108, 2021.

[30] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, Aarhus, Denmark, May 2005.

[31] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symposium on Security and Privacy, 2007. SP'07*, Berkeley, CA, USA, May 2007.

[32] L. Touati and Y. Challal, "Batch-Based CP-ABE with Attribute Revocation Mechanism for the Internet of Things," in *Proceedings of the International Conference on Computing*, IEEE, Shillong, India, 2015.